主要内容

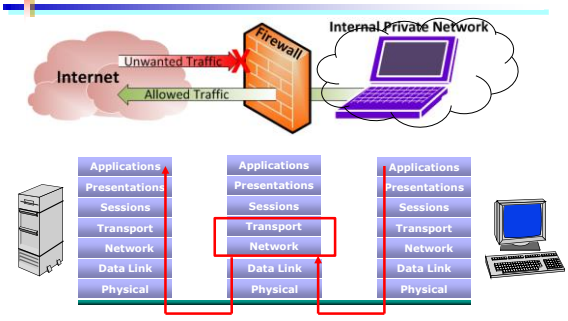
三、网络安全基础设施

◆防火墙的基本原理（linux的防火墙）

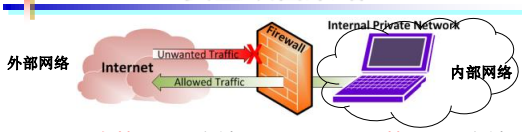
- ❖防火墙的类型
- ❖防火墙的部署（选）

◆大作业开题讨论

包过滤防火墙示意图



安全缺省策略



◆默认转发策略：没有被拒绝(deny)的流量都可以通过

- ❖黑名单，管理员必须针对每一种新出现的攻击，制定新的规则
- ❖方便用户，但安全性降低
- ❖用于开放的组织机构，如大学

◆默认丢弃策略：没有被允许(allow)的流量都要拒绝

- ❖白名单，比较保守
- ❖用于商业和政府机构

静态包过滤

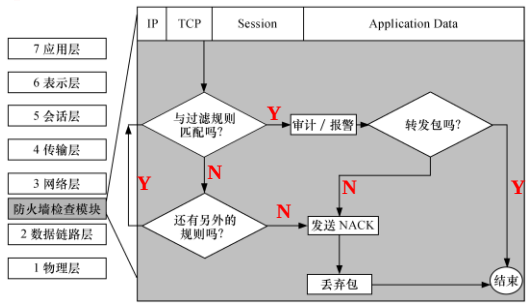
◆根据流经该设备的数据包地址信息决定是否允许该数据包通过

◆不检查数据区，只检查地址区

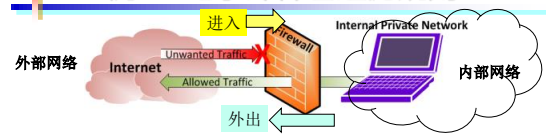
◆判断依据有(只考虑IP包)

- ❖ 数据包协议类型TCP、 UDP、 ICMP 、IGMP等
- ❖ 源、目的IP地址
- ❖ 源、目的端口FTP、 HTTP 、DNS等
- ❖ IP选项源路由、记录路由等
- ❖ TCP选项SYN 、ACK、 FIN 、RST等
- ❖ 其它协议选项ICMP、 ECHO、 ICMP、 ECHO 、REPLY等
- ❖ 数据包流向in或out
- ❖ 数据包流经网络接口eth0 eth1

静态包过滤防火墙模型



例1：电子邮件包过滤规则



- 针对电子邮件流量（SMTP协议）的包过滤规则集
  - 允许所有入站和出站的电子邮件流量
  - 禁止其他流量
- 说明：
  - SMTP协议：服务器使用TCP 端口25

防火墙规则

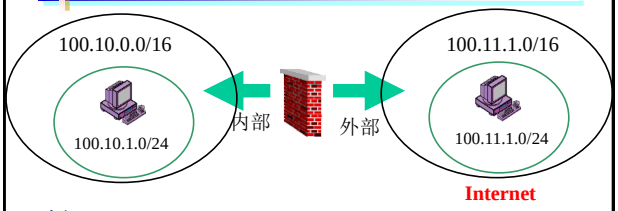
规则	包方向	源地址	目的地址	协议	目的端口	动作
1	入	外部	内部	TCP	25	允许
2	出	内部	外部	TCP	>1023	允许
3	出	内部	外部	TCP	25	允许
4	入	外部	内部	TCP	>1023	允许
5	either	any	any	any	any	拒绝

规则说明

- 每条规则的含义是
  1、允许流入的电子邮件流量
  2、允许对（1）的SMTP连接进行响应
  3、允许流出的电子邮件流量
  4、允许对（3）的SMTP连接进行响应
  5、默认规则（Deny）
- 这种规则设置有什么问题?
  - 增加约束：检查TCP状态标记；跟踪源端口号

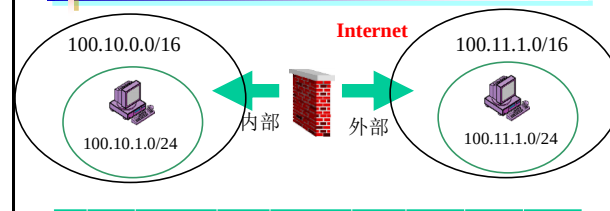
规则	包方向	源地址	源端口	目的地址	协议	目的端口	标记	动作
4	进入	外部	25	内部	TCP	>1023	ACK	允许

例2：子网包过滤规则



例：设网络100.10.0.0/16不愿意其它Internet主机访问其站点；但它的一个子网100.10.1.0/24和Internet上一个大学的实验室100.11.1.0/24有合作项目，因此允许该大学的实验室访问该子网，而不允许该大学的其他网段访问。

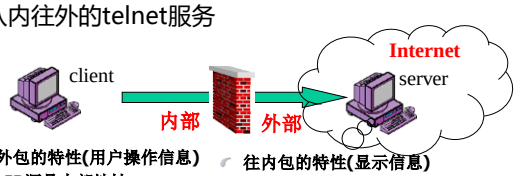
例2：子网包过滤规则



规则	包方向	源地址	源端口	目的地址	协议	目的端口	标记	动作
1	入	100.11.1.0/24	any	100.10.1.0/24	any	any	any	允许
2	出	100.10.1.0/24	any	100.11.1.0/24	any	any	any	允许
3	either	any	any	any	any	any	any	deny

例3：远程登录Telnet包过滤规则

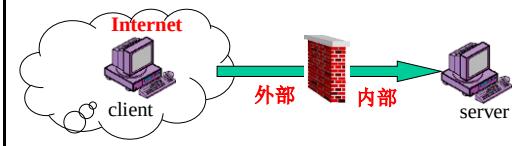
◆从内往外的telnet服务



- 往外包的特性(用户操作信息)
  - IP源是内部地址
  - 目标地址为server
  - TCP协议，目标端口23
  - 源端口>1023
  - 连接的第一个包ACK=0，其他包ACK=1
- 往内包的特性(显示信息)
  - IP源是server
  - 目标地址为内部地址
  - TCP协议，源端口23
  - 目标端口>1023
  - 所有往内的包都是ACK=1

例3：远程登录Telnet包过滤规则（续）

◆从外往内的telnet服务



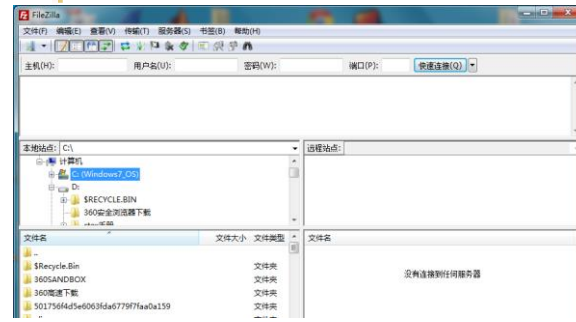
- 往内包的特性(用户操作信息)
  - IP源是外部地址
  - 目标地址为本地server
  - TCP协议，目标端口23
  - 源端口>1023
  - 连接的第一个包ACK=0，其他包ACK=1
- 往外包的特性(显示信息)
  - IP源是本地server
  - 目标地址为外部地址
  - TCP协议，源端口23
  - 目标端口>1023
  - 所有往内的包都是ACK=1

针对telnet服务的防火墙规则

规则	包方向	源地址	源端口	目的地址	协议	目的端口	标记	动作
1	出	内部	>1023	外部	TCP	23	*	允许
2	入	外部	23	内部	TCP	>1023	ACK	允许
3	入	外部	>1023	内部	TCP	23	*	允许
4	出	内部	23	外部	TCP	>1023	ACK	允许

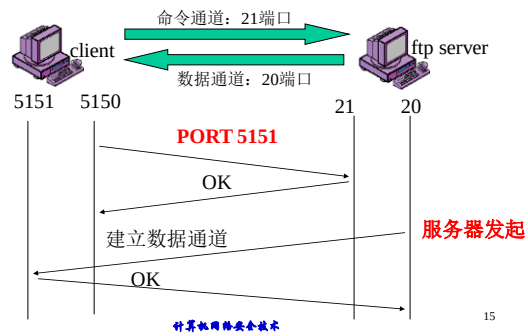
*: 第一个ACK=0, 其他=1

例4：文件传输FTP



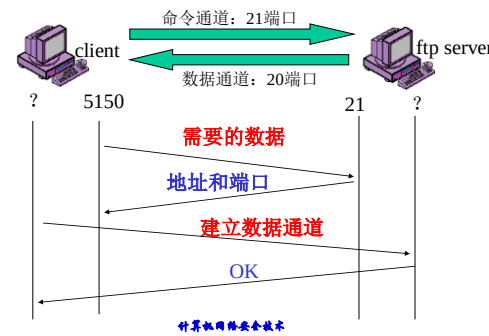
Ftp文件传输协议

主动模式



Ftp文件传输协议

被动模式



FTP协议处理

规则	包方向	源地址	源端口	目的地址	协议	目的端口	动作
1	in	外部	>1023	内部	TCP	21	allow
2	out	内部	21	外部	TCP	>1023	allow
3	out	内部	20	外部	TCP	>1023	allow
4	in	外部	>1023	内部	TCP	20	allow
5	out	内部	>1023	外部	TCP	21	allow
6	in	外部	21	内部	TCP	>1023	allow
7	in	外部	20	内部	TCP	>1023	allow
8	out	内部	>1023	外部	TCP	20	allow
9	both	*	*	*	*	*	deny

针对ftp的包过滤规则注意事项

- 建立一组复杂的规则集
- 动态监视ftp通道发出的port命令
  - 动态包过滤防火墙可以做到
- 注意
  - 静态包过滤防火墙比较适合于单连接的服务(比如telnet、smtp、 pop3等)，不适合于多连接的服务(比如ftp、IRC、H.323等)

静态包过滤防火墙

- 配置访问控制表
  - Access Control Lists (ACLs)
- 在网络层上进行监测
  - 并没有考虑连接状态信息
- 通常在路由器上实现
  - 实际上是一种网络的访问控制机制
- 特点：
  - 实现简单
  - 对用户透明
  - 效率高

静态包过滤防火墙的缺点

- 容易遭受IP地址欺骗
- 提供较低水平的安全性
- 缺少状态感知
- 创建规则比较困难，容易误配置
- 不支持用户身份认证

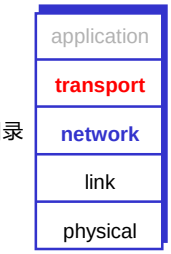
一些应对策略和规则

- IP地址欺骗攻击
  - 丢弃那些到达外部接口而源地址标记为内部主机地址的包
- 源路由攻击
  - 丢弃所有使用此选项的包
- 细小分段攻击
  - IP包的第一个分段必须包含最少的预定传输头。如果第一个分段被拒绝，则丢弃所有后继的分段

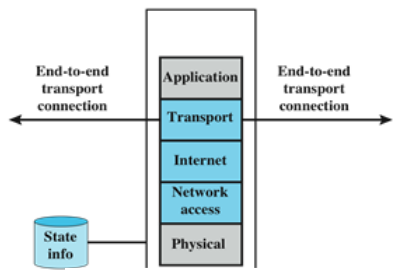
状态检测防火墙

Stateful Inspection Firewall

- 增加状态 **state**
- 工作在传输层
  - 建立出站（outbound）TCP连接目录
- ***可以记忆TCP连接，标志位等***
- 甚至可以记忆UDP包（如DNS请求）

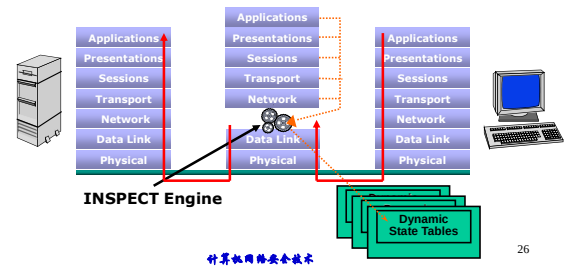


状态检测防火墙



状态检测技术（Stateful Inspection）

- 在数据链路层和网络层之间进行报文检测
- 建立用于维护**连接的状态表**



例：防火墙连接状态表

◆例：建立连接状态表

❖建立一个目录： outbound TCP connections

❖对入流量进行过滤

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.21.22.12	1046	192.168.1.6	80	Established

状态检测技术

“Stateful Inspection”

◆ 在包过滤的同时，检察数据包之间的关联性，数据包中动态变化的状态码

◆ 监测引擎：一个在网关上执行网络安全策略的软件模块

◆ 监测引擎采用抽取有关数据的方法对网络通信的各层实施监测，动态地保存起来作为以后执行安全策略的参考

◆ 状态监视器要抽取的数据

❖ 检查净荷数据区

❖ 可动态生成/删除规则

❖ 分析高层协议

技术标准

◆状态检测是防火墙的一个事实上的标准

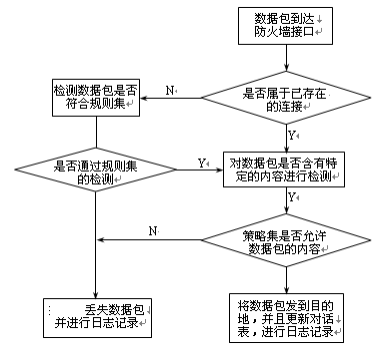
❖一个防火墙必须能跟踪和控制所有会话的**flow**

❖与原始的“包过滤”技术不同，状态检测分析流入和流出网络的“流flow”

❖可以基于通信会话信息（也可基于应用信息）做出实时的安全判断

❖ 这个结果通过跟踪穿越防火墙网关的通信会话的状态state和上下文来实现，不管这个连接connection 包含多么复杂的协议

工作过程



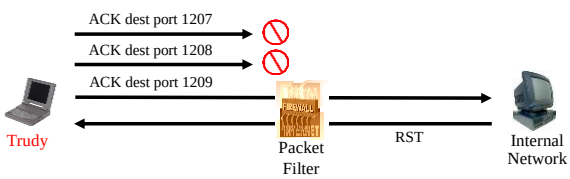
状态和上下文信息

- 状态和上下文信息
  - 数据包的头信息 (源地址、目的地址、协议、源端口、目的端口、包长度)
  - 连接状态信息 (哪一个连接打开了哪一个端口)
  - TCP 和 IP 分段数据 (例如：分段号、序号、窗口大小、其他TCP协议信息)
  - 数据包重组、应用类型、上下文校验 (即：包属于哪个通讯会话session)
  - 到防火墙的哪一个接口上
  - 从防火墙的哪一个接口上出去
  - 第二层信息（如VLAN ID号)
  - 数据包到达的日期和时间

例子：TCP ACK 扫描

- 端口扫描：攻击者扫描防火墙的开放端口
  - Port scanning is *first step* in many attacks
- 攻击者发送分组，ACK标志置位, 跳过三次握手（TCP的连接过程）
  - Violates TCP/IP protocol
  - ACK packet pass thru packet filter firewall
  - Appears to be part of an ongoing connection
  - RST sent by recipient of such packet

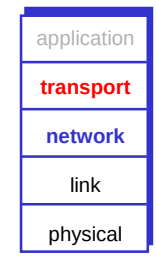
TCP ACK 扫描



- 攻击者得知 TCP1209端口可以通过防火墙
- 如何防止这种攻击?
  - 基于状态检测的包过滤防火墙

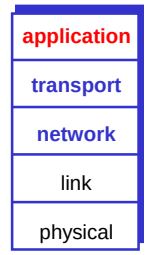
状态检测的包过滤防火墙的优缺点

- 在网络层上进行监测，并考虑连接状态信息
- 优点：
  - 安全性比静态包过滤防火墙高
  - 状态感知带来性能上的提高：保存连接状态，不需要繁琐的规则匹配，减少了访问控制规则的数量对性能的影响
- 缺点：
  - 工作在网络层和传输层，检查TCP和IP头，无法处理应用层数据，较低的安全性
  - 容易遭受IP欺骗攻击
  - 如果连接建立时没有遵循RFC建议的三次握手，会引入额外的风险，导致在DOS攻击时资源耗尽
  - 配置更复杂

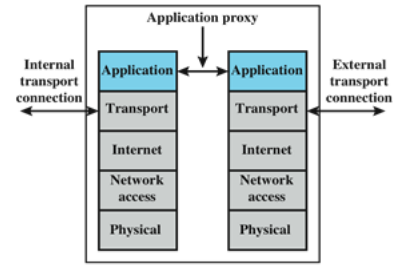


应用层网关 Application Proxy

- 也称为应用代理防火墙
  - A **proxy** is something that acts on your behalf
- 检查通过防火墙的应用层数据
- 特点
  - 可以监视包的内容
  - 可以实现用户认证
  - 所有的应用需要单独实现，检查和转发双向流量
  - 可以提供日志功能
  - 开销比较大



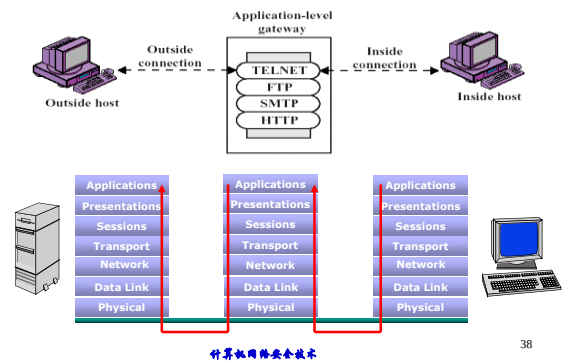
应用层网关 Application Proxy



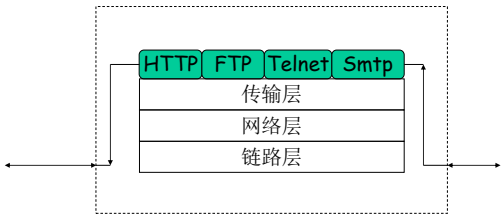
应用层网关的特点

- 当数据包通过防火墙时，创建一个新的数据包
- 代理具有完整的连接视图
- 攻击者必须与代理交互，才能使之转发信息
  - Attacker must talk to **proxy** and convince it to forward message
- 阻止某类攻击
  - 例：阻止端口扫描：Prevents some scans stateful packet filter cannot

应用层网关的结构示意图



应用层网关的协议栈结构

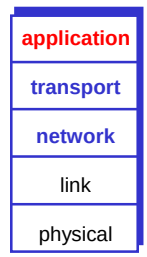


应用层网关实现

- 编写代理软件
  - 服务器软件
    - 但是它所提供的服务可以是简单的转发功能
  - 另一方面也是客户软件
    - 对于外面真正的服务器来说，是客户软件
  - 针对每一个服务都需要编写模块或者单独的程序
  - 实现一个标准的框架，以容纳各种不同类型的服务
    - 软件实现的可扩展性和可重用性
- 客户软件
  - 软件需要定制或者改写
  - 对于最终用户的透明性?
- 协议对于应用层网关的处理
  - 协议设计时考虑到中间代理的存在，特别是在考虑安全性，比如数据完整性的时候

应用层网关的优缺点

- 优点
  - 允许用户“直接”访问Internet
  - 具有连接和应用层数据的完整视图，易于记录日志
  - 可在应用层过滤恶意数据 (viruses, Word macros)
- 缺点
  - 新的服务不能及时地被代理
  - 每个被代理的服务都要求专门的代理软件
  - 客户软件需要修改，重新编译或者配置
  - 有些服务要求建立直接连接，无法使用代理
    - 比如聊天服务、或者即时消息服务
  - 代理服务不能避免协议本身的缺陷或者限制
  - 速度慢

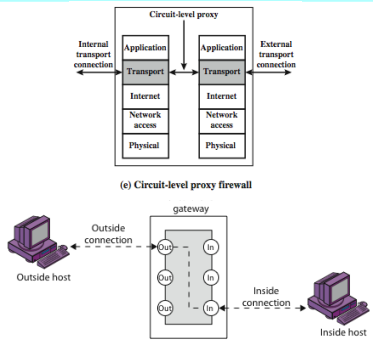


链路级网关Circuit Level Gateway

或：电路级网关，传输层代理协议

- 中继relay
  - relays two TCP connections
- 安全性
  - 建立两个TCP连接，不检查TCP报文段内容
  - 安全功能包括确定允许哪些连接
  - 信任内部用户发起向外的连接
  - 低开销
- 协议：SOCKS v5 （RFC1928）

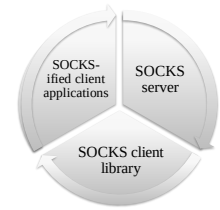
链路级网关Circuit Level Gateway



SOCKS Circuit-Level Gateway

- SOCKS v5： RFC1928
- 为C/S应用提供一个框架，使之能方便、安全地使用网络防火墙的服务
- 客户端访问 SOCKS 服务器
  - 认证authenticates, 发送中继请求 （relay request）
  - 服务器进行评估，决定是否建立连接。

组件



SOCKS 组件

- The SOCKS server
  - 运行在UNIX-based firewall、Windows 上
- SOCKS client library
  - 运行在受防火墙保护的内部主机上
- SOCKS-ified versions of several standard client programs
  - 例如，FTP，TELNET，HTTP

防火墙的局限性

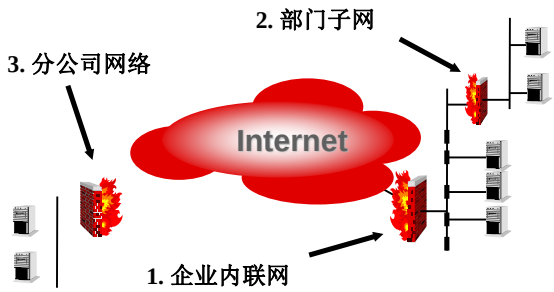
- 加密旁路
  - cannot protect from attacks bypassing it
  - eg utility modems, trusted organisations, trusted services (eg SSL/SSH)
- 内部威胁
  - cannot protect against internal threats
    - eg disgruntled or colluding employees
- 无线网络
  - cannot protect against access via WLAN
  - if improperly secured against external use
- 恶意软件
  - cannot protect against malware imported via laptop, PDA, storage infected outside

防火墙的部署

防火墙部署的位置

- 防火墙是软件或硬件设备的组合，可以位于：
  - 两/多个具有不同安全性要求的网络的互联之处，保护安全性要求高的网络
    - 最常见的是部署在本地网与Internet相连的地方，保护本地网
  - 网络边界之内，保护一小批特定主机
    - 例如部署在公司局域网内部的财务部门主机与局域网其它主机之间，保护财务部门敏感数据

防火墙的部署

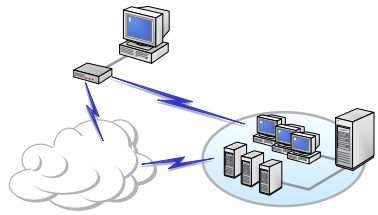


防火墙的部署

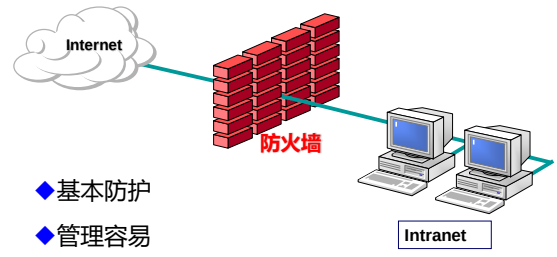
- 必须遵循的原则：
  - 所有的通信都经过防火墙
    - 所有在内部网络和外部网络之间传输的数据都必须通过防火墙
  - 防火墙只放行经过授权的网络流量
    - 只有被授权的合法数据，即防火墙系统中安全策略允许的数据，可以通过防火墙
  - 防火墙本身不会影响信息的流通，并能经受得起各种攻击

如何选择防火墙的网络结构

- ◆防御（堡垒）主机结构(Bastion Host)
- ◆3-Homed结构
- ◆多层次结构(Multi-Layered)

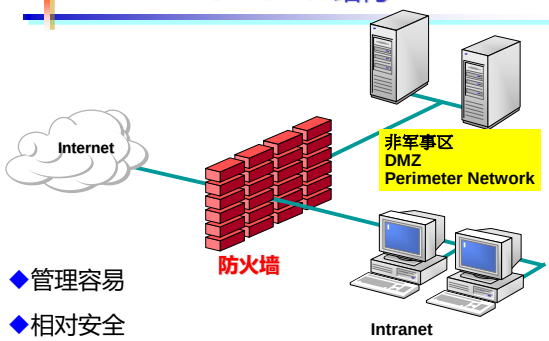


堡垒主机结构(Bastion Host)



- ◆基本防护
- ◆管理容易
- ◆成本低

3-Homed结构



- ◆管理容易
- ◆相对安全

非军事区DMZ

- ◆非军事区(DMZ)，也称为“隔离区”
  - ❖非安全系统与安全系统之间的缓冲区，把敏感的内部网络和其他提供访问服务的网络分开，阻止内网和外网直接通信，以保证内网安全。
  - ❖可以放置一些公用服务器，如Web服务器、FTP服务器、邮件服务器等。

DMZ网络访问控制策略

1.内网可以访问外网，防火墙需要进行源地址转换。

2.内网可以访问DMZ

3.外网不能访问内网

内网中存放的私有数据不允许外网的用户进行访问。

4.外网可以访问DMZ，由防火墙完成对外地址到服务器实际地址的转换。

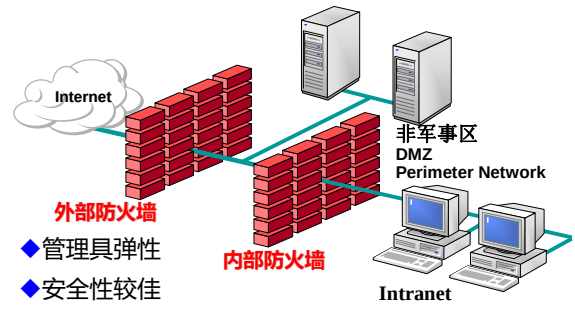
5.DMZ不能访问内网，

如果违背此策略，则当入侵者攻陷DMZ时，就可以进一步进攻到内网的重要数据。

6.DMZ不能访问外网

此条策略也有例外，比如DMZ中放置邮件服务器时，就需要访问外网，否则将不能正常工作。

多层次结构(Multi-Layered)



- 管理具弹性
- 安全性较佳
- 成本较高

各式防火墙结构的比较

	防御主机	3-Homed	多层次
安全性	低	中	高
管理的便利性	最容易	容易	具弹性
成本	低	中	高

防火墙的实现

补充

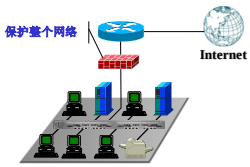
防火墙的实现

按形态分类

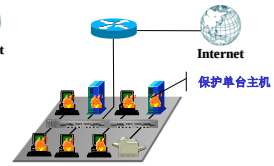
软件防火墙

硬件防火墙

按保护对象分类



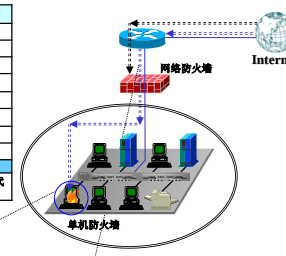
网络防火墙



单机防火墙

单机防火墙&网络防火墙

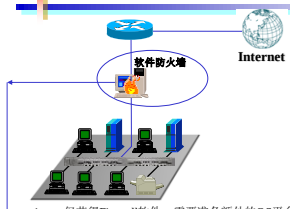
	单机防火墙	网络防火墙
产品形态	软件	硬件或者软件
安装点	单台独立的Host	网络边界处
安全策略	分散在各个安全点	对整个网络有效
保护范围	单台主机	一个网段
管理方式	分散管理	集中管理
功能	功能单一	功能复杂、多样
管理人员	普通计算机用户	专业网管人员
安全措施	单点安全措施	全局安全措施
结论	单机防火墙是网络防火墙的有益补充，但不能代替网络防火墙为内部网络提供强大的保护功能	



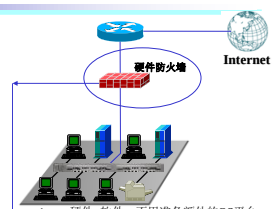
1. 保护单台主机
2. 安全策略分散
3. 安全功能简单
4. 普通用户维护
5. 安全隐患较大
6. 策略设置灵活

1. 保护整个网络
2. 安全策略集中
3. 安全功能复杂多样
4. 专业管理员维护
5. 安全隐患小
6. 策略设置复杂

硬件防火墙&软件防火墙



1. 仅获得Firewall软件，需要准备额外的OS平台
2. 安全性依赖低层的OS
3. 网络适应性弱（主要以路由模式工作）
4. 稳定性高
5. 软件分发、升级比较方便



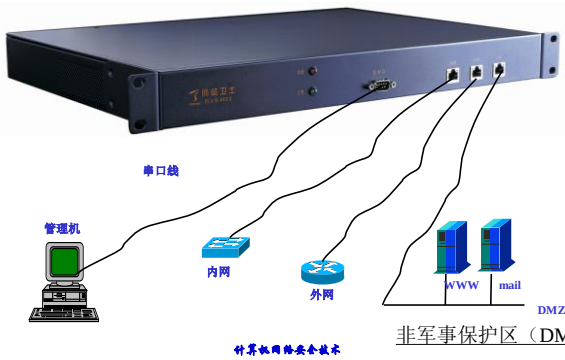
1. 硬件+软件，不用准备额外的OS平台
2. 安全性完全取决于专用的OS
3. 网络适应性强（支持多种接入模式）
4. 稳定性较高
5. 升级、更新不太灵活

	操作系统平台	安全性	性能	稳定性	网络适应性	分发	升级	成本
硬件防火墙	基于精简专用OS	高	高	较高	强	不易	较容易	Price=firewall+Server
软件防火墙	基于庞大通用OS	较高	较高	高	较强	非常容易	容易	Price=Firewall

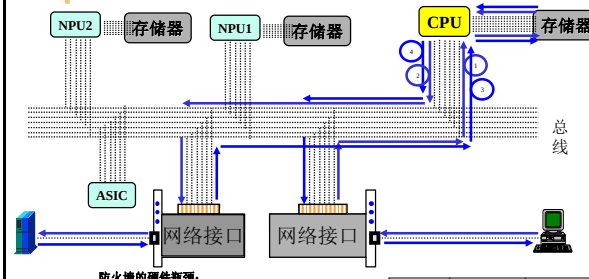
防火墙形态

1. 标准1U机箱
2. 配置n个 网络接口，内网、外网、DMZ（SSN） 三个接口固定，不可更改
3. 接口数量、类型不可更改
4. 国内标准220V交流电源输入，不需要额外的电源转换设备
5. 内存=64 M
6. 电源=AC90~260V，47~63Hz，0.15A / 0.25A
7. 主板采用集成化设计，稳定性、可靠性更高

防火墙连线图



防火墙单总线结构

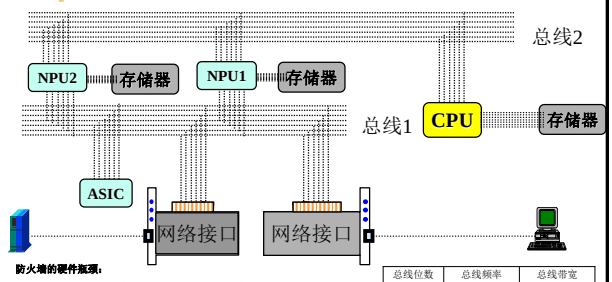


防火墙的硬件瓶颈：

- 处理器的计算能力：CPU+NPU+ASIC
- 总线带宽：设计多总线产品
- 存储器的存储速度
- I/O 接口速度

总线位数	总线频率	总线带宽
32 b	33M	1G
64 b	33M	2G
32 b	133M	4G
64 b	133M	8G

防火墙多总线结构



防火墙的硬件瓶颈：

- 处理器的计算能力：CPU+NPU+ASIC（提高主频或者采用多处理器）
- 总线带宽：设计多总线产品（提高总线带宽或者采用多总线结构）
- 存储器的存储速度
- I/O 接口速度

总线位数	总线频率	总线带宽
32 b	33M	1G
64 b	33M	2G
32 b	133M	4G
64 b	133M	8G

防火墙硬件结构种类

中低端

- CPU + 33M总线 + 10/100/1000M网络接口

中高端

- CPU + 133M总线 + 10/100/1000M网络接口
- CPU + ASIC + 133M总线 + 10/100/1000M网络接口
- CPU + NPU + 133M总线 + 10/100/1000M网络接口
- CPU + NPU + ASIC + 133M总线 + 10/100/1000M网络接口

电信级产品

- n个CPU + n个NPU + n个ASIC + n条总线 + n个10/100/1000

网络接口高处理能力　　逐步提高总线带宽　　逐步提高接口速率

1. 加密处理：采用 ASIC 芯片
2. 内容过滤：采用通用 CPU
3. 网络报文处理：采用 NPU

基于通用CPU的防火墙的特点

- 通用CPU的优点：高灵活性、高扩展性
- 通用CPU由于考虑了各种应用的需要，具有一般化的通用体系结构和指令集，容易支持复杂的运算并容易开发新的功能
- 随着通用CPU性能的快速提高，基于通用CPU防火墙的处理速度和能力将会大幅度提高，能够很好的适应多接口百兆、千兆防火墙的计算要求

基于ASIC加速技术防火墙的特点

- ASIC：Application Specific Integrated Circuit，专用集成电路
- ASIC作为硬件集成电路，它把指令或计算逻辑固化到硬件中，获得高处理能力，提升防火墙性能
- ASIC最大缺点是缺乏灵活性，指令或计算逻辑固化到硬件中，很难修改升级、增加新的功能或提高性能
- ASIC设计和制造周期长（设计和制造复杂ASIC一般需要花费12～18个月），研发费用高

基于NP加速技术防火墙的特点

- 网络处理器（Network Processor，简称NP）
- 能够直接完成网络数据包处理的一般性任务，如TCP/IP数据的校验和计算、包分类、路由查找等，同时，硬件体系结构的设计也弥补了传统IA体系的不足，它们大多采用高速的接口技术和总线规范，具有较高的I/O能力
- 基于NP的数据包处理能力得到了很大提升，很多需要高性能的领域，如千兆交换机、防火墙、路由器的设计都可以采用网络处理器来实现

Intel 公司第一代NP芯片

Linux中的包过滤防火墙Netfilter

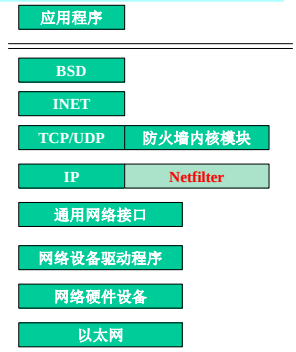
补充

Linux中的包过滤防火墙Netfilter

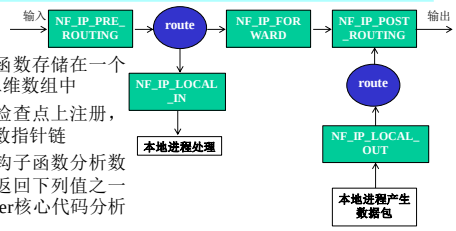
- Linux中的包过滤防火墙Netfilter
  - 包含在Linux 2.4以后的内核中，可以实现防火墙、NAT（网络地址翻译）和数据包的分割等功能。
  - netfilter工作在内核内部，而iptables则是让用户定义规则集的表结构
  - Netfilter/iptables从ipchains和ipwadfm（IP防火墙管理）演化而来，功能增强
- 功能
  - 包过滤：filter表不会对数据报进行修改，而只对数据报进行过滤
  - NAT：NAT表监听三个netfilter钩子函数：NF_IP_PRE_ROUTING、NF_IP_POST_ROUTING及NF_IP_LOCAL_OUT
  - 数据报处理：mangle表在NF_IP_PRE_ROUTING和NF_IP_LOCAL_OUT钩子中进行注册，对数据报的修改。

Netfilter在内核中的位置

- IP和防火墙内核模块之间
- IP和Netfilter结构相对独立
- 为每种网络协议（IPv4、IPv6、IPX等）定义一套**钩子函数（hook）**
- 数据报流经协议栈的几个检查点时被调用
- 对报文进行处理（修改、丢弃或传送给用户进程）
- 使用IPTables组件在用户空间对Netfilter进行设置，定制自己的防火墙



Netfilter检查点

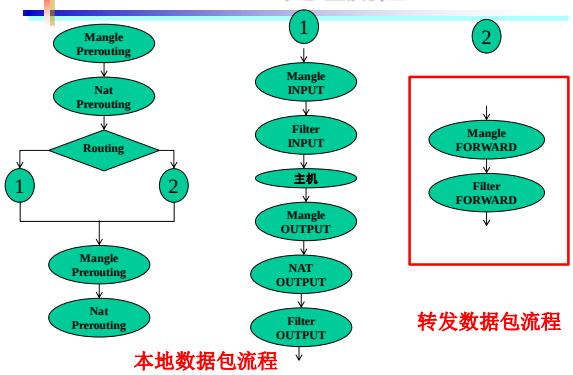


- 定义的钩子函数存储在一个list_head的二维数组中
- 钩子函数在检查点上注册，形成一条函数指针链
- 每个注册的钩子函数分析数据报结束后返回下列值之一，告知Netfiler核心代码分析结果
  - NF_ACCEPT：允许数据报文通过，进入下一步处理
  - NF_DROP：丢弃该报文
  - NF_STOLEN：由钩子函数处理该数据报，不再继续传送
  - NF_QUEUE：将数据报加入队列，交由用户程序处理
  - NF_REPEAT：再次调用该钩子函数

IPTables

- **Netfilter/IPTables**
  - iptables组件是用户空间（userspace）的工具，插入、修改和删除信息包过滤表中的规则。
  - Netfilter的钩子函数
  - 指导这些钩子函数如何工作的一系列规则
- **钩子函数通过访问表中的规则判断应该返回什么值给Netfilter模块**
- **内建表**
  - Mangle，Nat
  - Filter：NF_IP_LOCAL_IN、NF_IP_FORWARD、NF_IP_LOCAL_OUT三处组册了钩子函数
  - 注入模块的方式，调用Netfilter的接口函数创建新的表

Netfilter处理流程



本地数据包流程

转发数据包流程

动态包过滤例子

- Linux中的Netfilter
- **网络连接跟踪信息/proc/net/ip_conntrack**
  - Tcp 6 117 SYN-SENT src=192.168.1.6 dst=35.115.106.88 sport=32775 dport=22**[UNREPLIED]** src=35.115.106.88 dst=211.154.172.2 sport=22 dport=32775 use=2

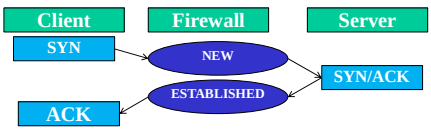
协议 | 协议号 | 生命期 | 状态

未收到任何响应

应答包信息

动态包过滤例子

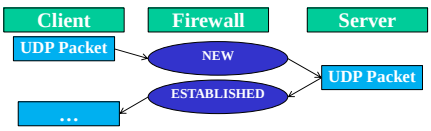
- TCP



- Tcp 6 117 **SYN-SENT** src=192.168.1.5 dst=192.168.1.35 sport=1031 dport=23**[UNREPLIED]** src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 use=2
- Tcp 6 57 **SYN-RECV** src=192.168.1.5 dst=192.168.1.35 sport=1031 dport=23 src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 use=1
- Tcp 6 631999 **ESTABLISHED** src=192.168.1.5 dst=192.168.1.35 sport=1031 dport=23 src=192.168.1.35 dst=192.168.1.5 sport=23 dport=1031 use=1

动态包过滤例子

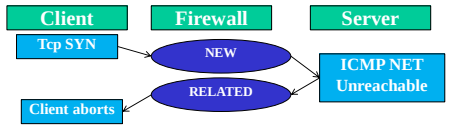
- UDP



- Udp 17 20 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025**[UNREPLIED]** src=192.168.1.5 dst=192.168.1.2 sport=1025 dport=137 use=1
- Udp 17 170 src=192.168.1.2 dst=192.168.1.5 sport=137 dport=1025 src=192.168.1.5 dst=192.168.1.2 sport=1025 dport=137 use=1
- Udp 17 175 src=192.168.1.5 dst=192.168.1.2 sport=137 dport=1025**[ASSURED]** src=192.168.1.5 dst=192.168.1.2 sport=1025 dport=137 use=1

动态包过滤例子

◆TCP和ICMP的关联



◆其他协议

- 类似于UDP，第一个包认为NEW，其后的应答包都是ESTABLISHED